Lab 7

Directions

- The labs are marked based on attendance and effort.
- It is your responsibility to ensure the TA records your progress by the end of the lab.
- Do each step of the lab in order.
- While completing these labs, you are encouraged to help your classmates and receive as much help as you like. Assignments, however, are individual work.
  You may not work on assignments during your lab time.
- If you complete the lab early, you should experiment with C++; however, you may leave if you prefer.
- If you do not finish the lab exercises during your lab time, you are encouraged to complete them later to finish learning the material. You will still receive full marks if you arrived on-time and put in your best effort to complete the lab.

1. Completing Lab 6

If you did not complete Lab 6, please do so now (you do not have to complete the challenge).

2. Getting rid of global variables

1. Create a new project for this lab, and copy the `reVariable.cpp` file provided on the web page into your project.
   - This file contains a small program which reads some information in from the user and displays it in a summary.
   - The problem is, it's written using a number of global variables. Read through the code and understand where each variable is used, and what it does.
2. Change the `reVariable.cpp` program to get rid of **all** global variables.
   - You should make at least one a static (local) variable.
   - You may need to change the function parameters.
   - Do not remove any functions, and the program should seem identical to the user (same functionality).
3. **Understanding:**
   - How to use local variables instead of global variables.
   - How can a local variable retain its value between function calls, but still not be accessible outside the function?

3. Making Functions

1. Copy the `makeFunctions.cpp` file into your project from the course web site. Run the program to see what it does.
2. You must use functions to break-down program into smaller pieces. Do not change what the program does (the output should be the same); just changing the program's structure.
   - Which parts should be their own function? Create between 4 and 6 functions: create functions for calculations and for repeated lines of code.
   - Here are some suggestions:
     - Each of the calculations for n^2, $1/n$, and $n!$ ($n! = 1 * 2 * ... * n$, for example $4! = 1 * 2 * 3 * 4 = 24$).
       Think about the arguments these functions will need, and their return types.
     - A couple different functions for outputting formatted data to the screen. Notice how there are a lot of lines like the following:
       `cout << setw(___) << right << ____;`
       Creating a few functions which replace all these `cout` statements in `main()`.
       What type of parameter(s) will these functions need to accept?
       Hint: name them like `outputOneInt(...)`, and so on.
   - Note that for this exercises you will be making smaller functions that you normally will, but it will give you a chance to practice.
3. Move all of your new functions below your `main()` function. You will need to add prototypes for each of your functions.
4. **Understanding:**
   - How to create a function which accepts one or two parameters, and returns a value.
   - The difference between having a function return a value, and output a value.
   - How to use prototypes to support re-ordering functions.

4. Sort function

1. Create a new program called `sort.cpp`.
2. Create a function which does the following:
   - Name it `sortTwoValues()`
   - Accepts two floating point values as parameters named `a` and `b`.
   - If `a` > `b`, then swap the values (i.e., the calling code will have the values changed).
   - Return a `bool` value: `true` if you had to swap, `false` if you did not swap.
   - Hint: How do you need to pass the data into the function in order to swap the values.
3. Place the following code in `main()` to use your `sortTwoValues()` function:


```
cout << "Enter two floating point numbers: ";
float val1, val2;
cin >> val1 >> val2;

cout << "Calling sortTwoValues() - returns: "
          << sortTwoValues(val1, val2)<<endl;
cout << "Ordered: (" << val1 << ", " << val2 << ")" << endl;
```

4. Test your function.
   - When you use the number 1 and 2, it should not swap them;
   - When you use the values 2 and 1 it should swap them.
   - Test with some other values.
5. **Understanding:**
   - Able to write functions which modify the parameters which are passed in.
   - Understand how to read code which uses pass-by-value and pass-by-reference.

5. Challenge – Not required

- Write a program encrypts and decrypts some text:
  - First, asks the user if they wish to encrypt or decrypt.
  - If encrypting, ask the user for a string (full line of text), and then output an encrypted version. You can choose any encryption method you like, an easy way is just add 4 to the value of each character. A more complex version is add 1 to the first character, 2 to the second, and so on. Or, use a more complex approach if you can!
    - Remember than you can access the characters of a string using []. Example:
      ```
      string str = "Hello";
      str[0] = str[0] + 4;
      ```
  - If decrypting, ask the user for the encrypted string and then decrypt it for them using a separate decrypting function. Your decrypting function must be the inverse of your encryption function (otherwise, the message is lost).
  - Make sure you use functions while creating your program. Have a function encrypt a single character, and another encrypt a full string, and likewise for decrypting.

6. Skills and Understanding

You should now be able to answer all the "understanding" questions in the previous sections. Complete the following to get credit for the lab:

- Show the TA the following:
  - Your operational programs which complete all of the above tasks.
  - The TA may ask you to explain any section of the lab, or answer any of the "Understanding" questions.
- **Nothing** is to be submitted electronically or in hard-copy for this lab.

